Weekplan: Introduction

Philip Bille Inge Li Gørtz

Reading

Introduction to Algorithms, Cormen, Rivest, Leisersons and Stein (CLRS): Chapter 1.

Exercises

1 Find Peaks Let $A = [2, 1, 3, 7, 3, 11, 1, 5, 7, 10]$ be an array. Solve the following exercises.

1.1 [w] Specify all peaks in A.

1.2 [w] Specify which peaks the two linear time algorithms find.

1.3 Specify the sequence of recursive calls the recursive algorithm produces. First assume the algorithm visits the left half of the array if both directions are valid. Afterwards specify all the possible sequences of recursive calls the algorithm can make when the algorithm can pick any of the two directions when they are both valid.

2 Valleys Suggest a *valley problem* as the opposite of the peak problem. Give a precise definition of the valley problem.

3 Algorithms and Applications

3.1 Pick a data structure from your introductory programming course and discuss it's strengths and limitations.

3.2 Suggest a real-world problem where only the optimal solution will do. Similarly, suggest a real-world problem where an approximate solution suffices.

3.3 Suggest relevant measures of complexity of algorithms other than time. Suggest at least 3.

4 Properties of Peaks Let A be an array of length $n \geq 1$. Solve the following exercises.

4.1 Prove that there always exists at least one peak in A.

4.2 What is the maximum number of peaks that can be in A?

4.3 Suppose we change the definition of peak as follows: $A[i]$ is a peak if $A[i]$ is *strictly larger* than it's neighbors. What is the effect on the above properties?

5 Peaks Solve the following exercises.

5.1 [†] Implement and test one of the two linear time algorithms for finding peaks.

5.2 [†] Implement the recursive algorithm for finding peaks (be careful not to go out of bounds)

5.3 Describe the worst case inputs for each of the three peak algorithms.

5.4 Design an iterative version of the recursive algorithm for finding peaks. Write the pseudocode for the algorithm.

5.5 Prove that the recursive algorithm always finds a peak. *Hint:* Define an appropriate invariant that is valid in each of the recursive calls and use induction.

6 2D Peaks Let M be an $n \times n$ matrix (2D-array). An entry $M[i, j]$ is a peak if it is no smaller than its N,E, S, and W neighbors (i.e. $M[i][j] \geq M[i-1][j]$, $M[i][j] \geq M[i][j-1]$, $M[i][j] \geq M[i+1][j]$ and $M[i][j] \geq M[i][j+1]$). We are interested in efficient algorithms for finding peaks in A. Solve the following exercises.

6.1 Give a simple algorithm that takes $O(n^2)$ time.

6.2 [$*$] Give an algorithm that takes $O(n \log n)$ time. *Hint:* Start by finding the maximum number in the center column and use this to solve the problem recursively.

6.3 [$**$] Give an algorithm that takes $O(n)$ time. *Hint:* Construct a recursive solution that divides M into 4 quadrants.